Assessing the Performance of Unreliable Computing Systems with Non-Exponential Task Time Distributions

Robert W. Rowan & Pierre M. Fiorini [*]
Department of Computer Science
University of Southern Maine
Portland, ME 04104
USA
{*rowan@cs.,pfiorini@*}*usm.maine.edu*

This paper presents a methodology for employing an analytic model to evaluate expected performance and dependability measures for an unreliable computing system comprised of any number of components executing any number of tasks where failure and repair rates are assumed to be exponentially distributed, but the task service times may be generally distributed. The model consists of a Markov Reward Model (MRM) constructed to incorporate the performance and durability characteristics of the system, i.e. transitions represent service, failure, and repair times. The model is then solved at each task completion point (or epoch boundary) via matrix analytic techniques. Various performance and dependability measures can thereby be calculated at those epoch completion points embedded in the task stream, providing insight into the instantaneous and cumulative behavior of the system.

One well-known issue with the MRM aproach is the very large state-space required to accurately analyze complex computing systems. The technique described in this work addresses the state-space problem in two ways, using an epoch approach and taking advantage of symmetry in the system structure and task stream. We utilize an epoch approach by analyzing each completion epoch independently, as illustrated in Figure 1. We calculate the distribution and expected values of various metrics for each epoch, then sum over all epochs to get the cumulative metrics for the entire job. This approach significantly reduces the state space as it requires that only the state information of the *current* epoch be stored. Using this approach, we can calculate the performance measure task completion time, the dependability measures system Availability, time to failure (MTTF), time between failures (MTBF), and time to repair (MTTR), and the performability measure Work done by the system.

Our base system can be thought of as a parallel system of P identical processing elements as shown in figure 2. The base workload on the system is comprised of N iid tasks which are all present at time $x = 0$. The service time of each task has a *matrix exponential* (ME) representation given by an m-dimensional vector-matrix pair, $< \mathbf{p}, \mathbf{B} >$, such that its *Cumulative Distribution Function*, CDF, is given by

$$F(x) = Pr(X \leq x) = 1 - \mathbf{p} exp(-x\mathbf{B})\epsilon', \tag{1}$$

where ϵ' is an m-dimensional column-vector of ones. A CDF with an m-dimensional ME representation can be said to be an m-phase distribution. It is widely accepted that any CDF can be represented (arbitrarily closely) by some ME representation.

The behavior of such a system depends upon the state of each of the system components; each of the components may be idle, busy, or failed (i.e. down or off). Our base model can be extended to include multi-tiered systems or those where the components may run in some degraded mode of operation. Some other possible extensions to the base model are discussed in the Conclusions section.

When discussing non-exponential distributions, the resumption of tasks whose processors failed during their execution becomes significant. This work explores two policies for resumption of tasks, preemptive resume ($prs \cong$ the resuming task time is sampled from a random variate (r.v.) of the same phase as its previous (aborted) execution) and preemptive repeat different ($prd \cong$ the resuming task time is sampled as a new r.v. from any phase of the distribution). The preemptive resume scenario requires a much larger state space as we must store information pertaining to the execution phases of previously failed tasks.

[*] This research is supported by NASA (EPSCoR Grant EP-02-12).

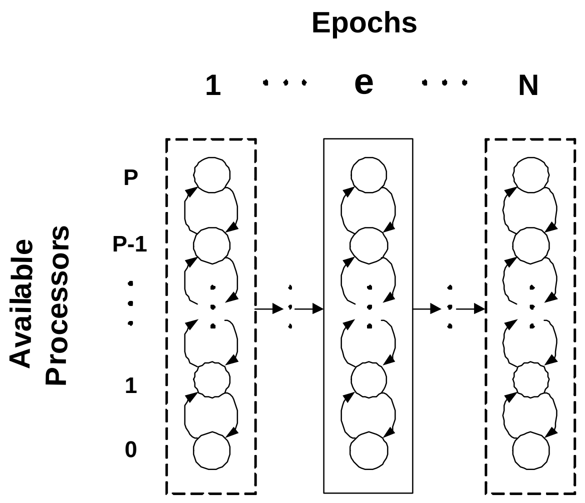

Figure 1: Metrics are calculated for each epoch $1 \leq e \leq N$, where the task completions are used as embedding points for the epoch boundaries.

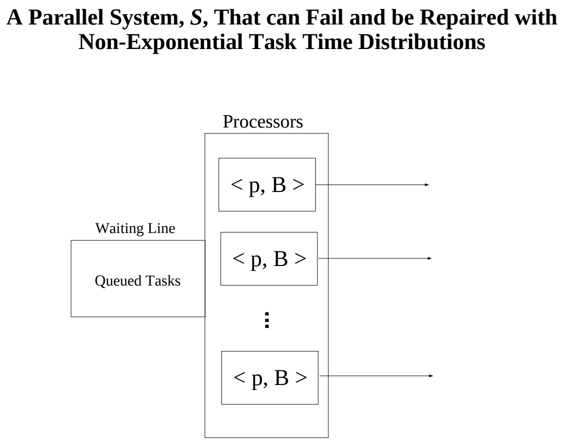
A Parallel System, S, That can Fail and be Repaired with Non-Exponential Task Time Distributions
Processors
Waiting Line
Queued Tasks
< p, B >
< p, B >
< p, B >

Figure 2: This figure shows our base model, a parallel processing system with non-exponential task time distributions where failures and repairs can occur.

A number of researchers have observed that many computer system related phenomena do not exhibit the characteristics of well-behaved statistical distributions. Since we would like to analyze real-world systems, we explore utilizing ME representations of the *Truncated Power-tail* distribution, TPT, to approximate distributions with infinite moments, i.e. distributions that are *not* well-behaved.

1 Our Approach

Utilizing ME functions and Linear Algebraic Queueing Theory (LAQT), we can define a vector-matrix pair, $< \wp(\ell), \mathcal{B}(\ell) >$, to characterize the distribution of the completion time for the ℓ^{th} epoch. Where $\wp(\ell)$ is the entrance vector and $\mathcal{B}(\ell)$ is the infinitesimal generator matrix of the completion process for the ℓ^{th} epoch. Given that the system is in vector state $\wp(\ell)$ at the beginning of epoch ℓ (time $x_\ell = 0$), then $\wp(\ell) exp(-x_\ell \mathcal{B}(\ell))\epsilon'$ tells us the probability that the epoch has not completed by time x_ℓ.

We construct $\mathcal{B}(\ell)$ from the underlying Markov chain and service time distribution. Diagonal elements, $[\mathcal{B}(\ell)]_{ii}$, are all > 0 and their quantity provides the rate at which the system leaves state i. Off-diagonal elements, $[\mathcal{B}(\ell)]_{ij}, i \neq j$, are all ≤ 0 and the absolute value gives the rate at which the system transitions to state j, given that the system is in state i. Note that $\sum_k [\mathcal{B}(\ell)]_{ik}$ is ≥ 0 for all rows and gives the rate at which the epoch completes, given the system is in state i.

The size of the state space often becomes the limiting factor in the size of the systems which we are able to accurately model. The epoch number, ℓ, tells us the number of tasks remaining in the system. We combine this information with that garnered from the state space to capture the instantaneous performance of the system. Hence, the state space must tell us how many processing elements (PEs) have and have not failed so that we can determine how many PEs are busy, how many are idle, and how many are down. The state must also tell us how many tasks are executing in each phase of the ME distribution. Note that if we do not assume that the PEs are identical and the tasks are iid, we may need to quantify either exactly which task is operating on which processor, exactly what phase each processor is executing in, or both. Such a requirement significantly increases the size of the state space. In the prs task resumption scenario, we must additionally use the state space to store information about the phase in which a failed task had been executing prior to failure.

The inverse of the $\mathcal{B}(\ell)$ matrix is the service time matrix, $\mathcal{V}(\ell)$. Elements $[\mathcal{V}(\ell)]_{ij}$, represent the amount of time during the ℓ^{th} epoch that the system spends in state j, given that the system began the epoch in state i.

In order to calculate the epoch entrance vector, $\wp(\ell)$, we must first calculate the *conditional probability transition matrix*, $\mathcal{Y}(\ell)$, as

$$\mathcal{Y}(\ell) = \mathcal{V}(\ell)\mathcal{M}(\ell)\mathcal{Q}(\ell)\mathcal{R}(\ell),$$

where $\mathcal{M}(\ell)$ is the *Transition Rate Matrix.* It is a diagonal matrix with $[\mathcal{M}(\ell)]_{ii} = [\mathcal{B}(\ell)]_{ii}$ for all i and all other elements are zero. $\mathcal{Q}(\ell)$ represents the state transition on an epoch completion and $\mathcal{R}(\ell)$ represents the state transition due to a task arrival from the queue. Their construction depends upon the modeling situation and the task distribution.

We can usually garner the initial entrance vector, $\wp_I = \wp(1)$, from the modeling situation and can calculate $\wp(\ell)$ by the following:

$$\wp(\ell) = \wp(\ell - 1)\mathcal{Y}(\ell - 1).$$

Finally, we can compute $T(\ell)$, the mean time to complete the ℓ^{th} epoch,

$$T(\ell) = \wp(\ell)\mathcal{V}(\ell)\epsilon',$$

and T_N, the mean time to finish all N tasks,

$$T_N = \sum_{\ell=1}^{N} T(\ell).$$

Similarly, Work and Availability can be calculated by using additional matrix operators. MTTF, MTBF, and MTTR can be calculated by creating an absorbing state (or sink) and running the calculations until the system completely (within some threshold) enters the absorbing state.

2 Results

Figure 3 shows the mean job completion time for systems where the number of tasks initially in the system, N, is equal to the number of processing elements, P, a *prd* resumption policy, exponentially distributed failure and repair times with repair rate, $\beta = 1.0$ and a variable failure rate, α. The task service times are sampled from a three-phase hyperexponential with a mean of 1.21 and coefficient of variation equal to 2.62. We see that the as the failure rate increases, performance (as measured by the job completion time) *improves.* This counter-intuitive phenomenon is a result of the task resumption policy. Longer tasks are more likely to fail before completion and when a failed task is restarted, the new service time is sampled as a new r.v. For either resumption scenario, it is likely that the new service time will be shorter than the original, and in the *prd* case, the new service time is likely to be significantly shorter.

Figure 4 shows the epoch completion times for a system comprised of two processors executing ten tasks with a *prs* resumption policy, $\beta = 1.0$, and variable α. The task service times are sampled from a two-phase hyperexponential with a mean of 1.0 and a coefficient of variation equal to 3.16. As can be seen from this graph, performance does *not* improve with increasing failure rate, in contrast with the *prd* resumption policy, discussed above. We observe three distinct regions in our epoch completion time results, an initial transient region where short tasks finish quickly, a steady-state region where task completion times gradually increase, and a final transient region where expected task times significantly increase due to the likely presence of large tasks in the system.

3 Conclusions

Our goal is to accurately model real world computing systems and analytically solve for performance and dependability measures and this work is a significant step towards that goal. In this work, we have made certain assumptions that have imposed limitations on the systems to which we may apply this technique, however the base model can be adapted to overcome many of these restrictions.

The model, as presented, only allows failures and repairs to occur independently and sequentially, but only minor modifications are required to accomodate failures or repairs that occur in batches and/or all at

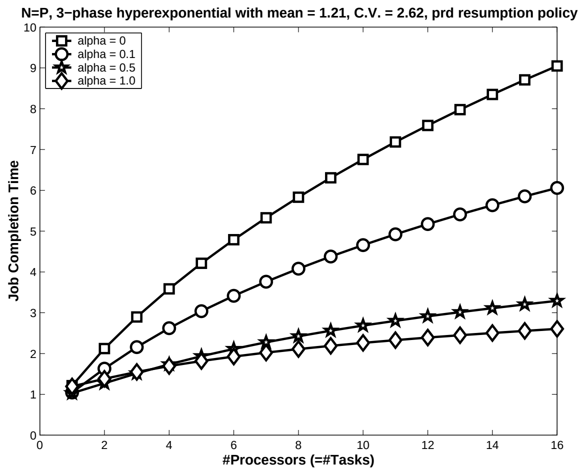

Figure 3: Expected job completion time results for a system with a *prd* task resumption policy.

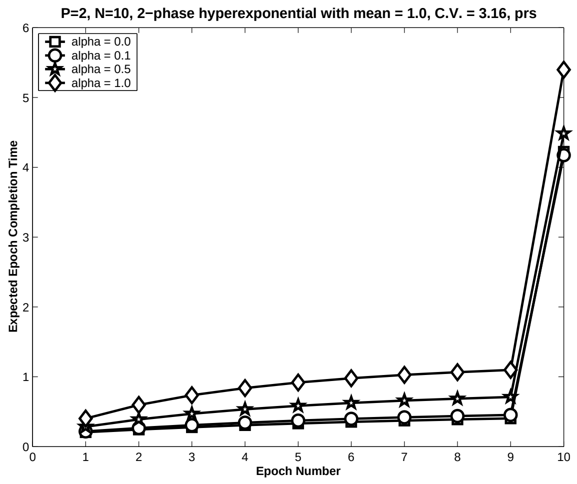

Figure 4: Expected task completion time results for a system with a *prs* task resumption policy.

one time. The base model imposed the restriction that all the tasks comprising the workload be present at time $x = 0$. In other words, the workload is completely predefined and there can be no task arrivals. This limitation can readily be overcome by using a form of busy period analysis. We additionally assumed that all tasks were asynchronous. However, we feel that by using a general task service-time distribution, we should be able to accurately capture (or at least closely approximate) the blocking behavior of synchronous tasks.

One obvious ommission from our resumption scenarios is a repeat identical scenario, such that once a failed task resumes, it executes for at least the amount of time for which it had been executing prior to its initial failure. We might approximate this behavior by incorporating some of the principles of the *prs* model, specifically storing previous system behavior in the state space, or with some general distribution, such as a hypoexponential, that requires sequential phase execution. On the whole, we feel that this technique shows significant promise toward achieving the goal of predicting performance and durability measures for any computing system.

References

Bobbio, A. and Trivedi, K. (1990). Computation of the Distribution of the Completion Time When the Work Requirement is a PH random Variable. *Communications in Statistics- Stochastic Models 6*(1), 133-150.

Crovella, Taqqu, and Bestavros (1998). Heavy-Tailed Probability Distributions in the World Wide Web. In *A Practical Guide To Heavy Tails*, pp. 3-26. New York: Chapman and Hall.

Fiorini, P., Campbell, C., and Lipsky, L. (2002). An Analytic Approach to Assess the Performability of Parallel & Distributed Systems. In Proc. of *The* 15^{th} *International Conference on Parallel and Distributed Computing Systems (PDCS-2002).* Louisville, KY.

Lipsky, L. (1992) *Queueing Theory: A Linear Algebraic Approach.* New York: MacMillan and Company.

Neuts, M.F. and Lucantoni, D.M. (1979). A Markovian Queue with N Servers Subject to Breakdowns and Repairs. *Management Science 29*(9), 849-861.

Rowan, R.W., Fiorini, P., and Lipsky, L. (2003). Assessing the Perfomability of Computing Systems Subject to Failure and Repair Using Non-Exponentially Distributed Task Times. In Proc. of *The 2003 International Multiconference in Computer Science and Engineering,* pp. 1363-1369. Las Vegas, NV.